B.I.A.G.R.A.

BIbliotecA de proGRamación científicA

José Angel de Bustos Pérez
<jadebustos@mixmail.com>

Versión 1,0, 5 de noviembre de 2015.
LaTeX 2ε

Índice general

Índice de cuadros

Capítulo 1

¿Que es *B.I.A.G.R.A* ?

- *B.I.A.G.R.A* es una BIbliotecA de proGRamación científicA programada enteramente en **C** y pensada para ser utilizada en programas escritos en **C**.
- *B.I.A.G.R.A* ha sido desarrollada y testeada bajo *LiNUX*, por lo cual sus características se adaptan a las de este sistema operativo.
- *B.I.A.G.R.A* es de libre distribución y su autor no se hace responsable de su mal uso y/o de sus posibles fallos.

1.1. ¿Lenguaje C?

Alguien se puede preguntar el motivo por el cual he elegido el Lenguaje **C** y no *FORTRAN*, lenguaje tradicional para la programación científica.
Los motivos son varios siendo el fundamental la potencia, la modularidad de **C** y el hecho de ser un lenguaje de proposito general.

Las razones que alegan la inmesa mayoría de programadores de *FORTRAN* sobre las ventajas que tiene *FORTRAN*:

1. Es un lenguaje facil de usar.
2. Casi todo el mundo utiliza *FORTRAN*, en este tipo de programación.
3. Hay mucho código escrito en *FORTRAN*.

Aplicando esta forma de pensar bien podríamos pensar que el *sol gira sobre la tierra*, ya que hace muchos años la mayoría de la gente estaba de acuerdo con esta idea.

Las razones por las que fundamento la elección de **C** son:

1. Es un lenguaje potente y que genera un código de muy rápida ejecución.
2. Es un lenguaje flexible.
3. Es un lenguaje estructurado.
4. Es un lenguaje de proposito general.
5. El código escrito en **C** es portable y la depuración y mantenimiento de programas es "sencilla"[1].
6. Tiene una gran potencia en el manejo del hardware, lo que implica una gran potencia en el manejo de modos gráficos[2].

1.2. Algunas ideas generales sobre *B.I.A.G.R.A*

Lo primero que debemos destacar es que esta biblioteca esta pensada para programadores de **C** y ha sido desarrollada y testeada bajo **LiNUX**, luego en el resto de este documento se supondra que el lector esta utilizando **LiNUX** o cualquier otro sistema tipo **UNIX**.

Esta biblioteca esta pensada para resolver problemas generales, no para resolver tipos particulares de problemas, es decir, esta pensada para realizar programas, que, interactuando con el usuario puedan resolver un problema sin tener que modificar el código fuente cuando varien las condiciones del problema.

Por ejemplo, escribir un programa para invertir matrices, de cualquier orden, en vez de escribir un programa para invertir matrices de orden n y cuando se quiera invertir una matriz de un orden $m \neq n$ sea necesario cambiar el código fuente y recompilar el programa.

Para lograr esto se han utilizado *punteros*, es decir se ha evitado, en la medida de lo posible, el uso de *arrays*.

[1] Si el código esta bien estructurado y se ajusta al *ANSI C*.
[2] Deseable para según que aplicaciones.

Cuando hablemos de *vectores* nos estaremos refiriendo a un *puntero*, normalmente a datos del tipo *double*, para el cual se ha reservado memoria para contener n datos. Del mismo modo cuando nos refiramos a *matrices* estaremos hablando de matrices de *punteros*, también a datos del tipo *double*(normalmente), para las cuales se ha reservado memoria para contener $n * m$ datos.

En aquellos problemas que se utiliza un conjunto de varios datos se ha optado por agruparlos dentro de una estructura[3] para mayor comodidad.

En aquellos problemas en los que se pueda cometer algún error como por ejemplo:

- Fallo en asignaciones de memoria.
- Divisiones por cero.
- ...

la función que resuelve ese problema devolverá un entero indicando cual fue el estado en el que se terminó la ejecución de la función.

Cuando una función devuelva un dato, que no sea un código que indique el estado en el que se terminó la ejecución de la función, se indicará anteponiendo un prefijo al nombre de la función, el cual indicará que tipo de dato devuelve, por ejemplo:

1. *double dblFuncion(...)* función que devuelve un dato de tipo *double*.
2. *int intFuncion(...)* función que devuelve un dato de tipo *int*.
3. *double *dblPtFuncion(...)* función que devuelve un dato de tipo puntero a *double*.
4. *void Funcion(...)* función que no devuelve ningún dato.

[3]Declarada mediante typedef.

1.3. Códigos devueltos por las funciones

No es obligatorio que las funciones devuelvan un valor.
Cuando una función devuelva un valor será por dos razones:

- Para devolver el resultado de una operación.
- Para informar de como terminó una operación.

Es este último caso el que nos incumbe.

Cuando una función devuelva un dato indicando como terminó una determinada operación, este dato será, necesariamente, un entero y los códigos devueltos los podemos ver en la página **??**.

1.4. Como instalar *B.I.A.G.R.A* en LiNUX

Para instalar *B.I.A.G.R.A* lo primero que hay que hacer es entrar en el sistema como **root** y situarse en el directorio donde esten los fuentes de la biblioteca.

1.4.1. Intalación de la biblioteca *B.I.A.G.R.A* estática

Para instalar este tipo bibilioteca se puede hacer de dos formas:

1. *./instalar estatica*
2. *make estatica*

En realidad ambas hacen lo mismo, la opción con *make* en realidad ejecuta *./instalar estatica*.

Al realizar cualquiera de estas dos opciones se copiarán los ficheros cabezera a ***/usr/include/biagra*** y a continuación se creará la biblioteca ***/usr/lib/libbiagra.a***.

Luego si se utiliza una función de esta biblioteca, cuyo prototipo está en el fichero de cabecera *rngkutta.h* habrá que incluir en las directivas al prepocesador **#include <biagra/rngkutta.h>**.

1.4.2. Intalación de la biblioteca *B.I.A.G.R.A* dinámica ELF

1.4.3. Instalación de ambas bibliotecas

Para instalar la biblioteca *B.I.A.G.R.A* en su forma estática y dinámica ELF:

make todo

Esto lo que hace es ejecutar primero

./instalar estatica

lo cual instalará la biblioteca estática, y luego

./instalar elf

lo cual instalará la biblioteca dinámica ELF.

1.5. Como utilizar *B.I.A.G.R.A* en LiNUX

B.I.A.G.R.A es una biblioteca para programación científica, desarrollada para ser usada en programas escritos en **C**, se distribuye en varios formatos:

Biblioteca estática

Biblioteca dinámica ELF

1.5.1. Biblioteca estática

Para el uso de esta biblioteca hay que indicarle al *montador* que biblioteca debe *enlazar*. Por ejemplo, supongamos que hemos escrito un programa para resolver una ecuación diferencial por un método *Runge-Kutta* y hemos utilizado funciones cuyos prototipos estan en **edo.h** y **rngkutta.h**, si nuestro programa es *programa.c*, para crear el ejecutable:

*gcc programa.c -o programa -l**biagra** -l**m***[4]

[4] *B.I.A.G.R.A* utiliza la biblioteca estandar matemática.

1.5.2. Biblioteca dinámica ELF

1.6. Novedades en la versión 1,0

Pues todo ya que es la primera versión.

1.7. Condiciones de uso

1. El autor no se responsabiliza de su mal uso, de su posibles fallos[5], ni de las modificaciones que sean realizadas a dicha biblioteca.
2. Es una biblioteca de libre distribución.
3. Se puede modificar y añadir código, pero nunca distribuirlo bajo el nombre de *B.I.A.G.R.A* , en caso de modificación de la biblioteca se deben indicar aquellas funciones que fueron modificadas o añadidas.

1.8. Sobre el autor

Si alguien desea ponerse en contacto con el autor de esta biblioteca:

jadebustos@mixmail.com

Si tienes alguna sugerencia, pregunta, o has encontrado algún error en la biblioteca ya sabes donde encontrarme.

[5]Si los hubiera o hubiese.